ITRI 615

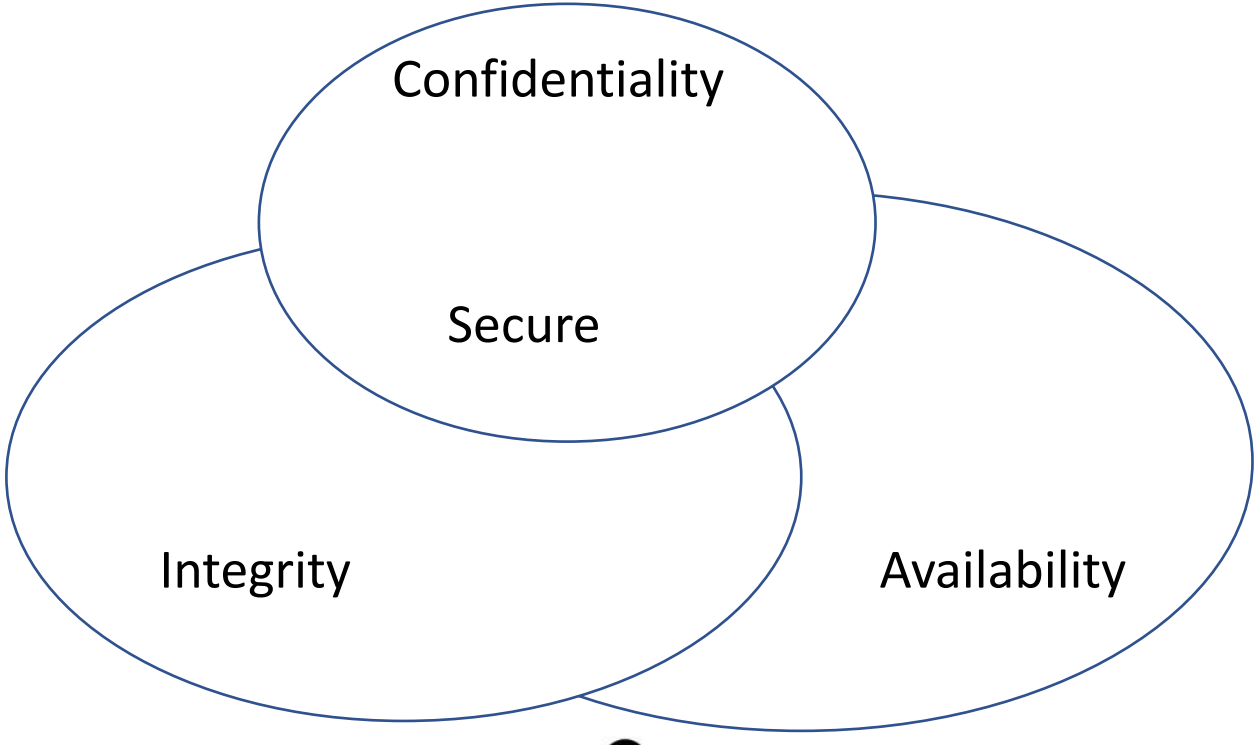
Pfleeger Chapter 1

Whitman & Mattord Chapter 1 & 2

Is there a security problem in computing?

- When do we consider something to be secure? Gold vs. data?
- For the duration of the course, a computing system will refer to the collection of hardware, software, storage media, data and people that an organization uses to perform computing tasks.
- Any system is most vulnerable at its weakest point. This leads to the principle of easiest penetration. It states that any computing system is only as strong as its weakest point.
- The four main types of threats include interception, interruption, modification and fabrication.

Objectives of computer security



Confidentiality

- Ensures that computer-related assets are accessed only by authorized parties.

- Access does not only refer to reading, but also viewing, printing, or even just knowing that a particular asset exists.

- Synonyms include secrecy and privacy.

Integrity

- Means that assets can be modified only by authorized parties or only in authorized ways.

- Modification refers to writing, changing, changing status, deleting and creating.

- It also includes error detection and correction, whereby errors, whether intentional or by accident, are identified and corrected.

Availability

- Means that assets are accessible to authorized parties at appropriate times. I.e. If a system or person has legitimate access, that access should not be prevented.

- A data item, service or system is available if

1. There is a timely response to our request.
2. Resources are allocated fairly.
3. The service or system involved follows a philosophy of fault tolerance.
4. The service or system can be used easily.
5. Concurrency is controlled.

Threats

- A threat to a computing system is a set of circumstances that has the potential to cause loss or harm.
- A threat is blocked by control of a vulnerability (weakness).
- A threat can refer to either the hardware component, the software component, the data component, or to a combination of the components.
- Threats are either human-initiated, computer-initiated, or nature-initiated.

Threats *cont.*

- The four main types of threats include:

1. Interception, where an unauthorized party has gained access to an asset.

2. Interruption, where an asset of the system has become lost, unavailable, or unusable.

3. Modification, where an unauthorized party not only accesses but tampers with an asset.

4. Fabrication, where an unauthorized party creates counterfeit objects on the computing system.

Computer criminals

- Amateurs. Ordinary computer professionals who discover they have access to something valuable.
- Crackers or malicious hackers. People that attempt to access computing facilities for which they have not been authorized.
- Career criminals. People that understand the targets of computer crime, and typically trade in companies' or individuals' secrets.
- Terrorists. People that use computers for:

1. Targets of attack.
2. Propaganda vehicles.
3. Methods of attack.

Harm

- Harm occurs when a threat is realized against a vulnerability.
- We can seek to:

1. Prevent it.
2. Deter it.
3. Deflect it.
4. Detect it.
5. Recover from its effects.

Encryption as control

- Encryption is the formal name for the scrambling process.
- Normal data, called cleartext, is taken and scrambled so that they are unintelligible to the outside observer. This takes place according to some encryption algorithm. Decryption takes place through an accompanying decryption algorithm back to its original state.
- Encryption does however not solve all computer security problems, and other tools should complement its use.
- It can be argued that weak encryption is worse that no encryption at all, as it gives users a false sense of security.

Software controls

- Programs are typically the second facet of computer security, after encryption, and includes:

1. Internal program controls. Parts that enforce security restrictions.

2. Operating system and network system controls. Limitations enforced by the operating system or network.

3. Independent control programs. Application programs that protect against certain types of vulnerabilities.

4. Development controls. Quality standards for program development.

Hardware controls

- Numerous devices have been created to assist in providing computer security, and include:

1. Hardware or smart card implementations of encryption.
2. Locks or cables limiting access or deterring theft.
3. Devices to verify users' identities.
4. Firewalls.
5. Intrusion detection systems.
6. Circuit boards that control access to storage media.

Policies and procedures as control

- Sometimes policies and procedures can be relied upon between users instead of hardware and software means.
- Something as simple as frequent password changes can have tremendous effect with no cost involved.
- Training and administration are key in reinforcing the importance of security policy.
- Legal and ethical controls are an important part of computer security, but unfortunately computing is progressing much faster than law.

Physical controls

- Some of the easiest, most effective, and least expensive controls are physical controls.
- Some examples include locks on doors, guards at entry points, backups, and physical site planning to minimize the risk of natural disasters.
- Although physical controls can have a significant impact on computer security, they are often overlooked.

Effectiveness of controls

- Controls are of no effect if they are not used properly. Several aspects that can enhance the effectiveness of controls include:

1. Awareness of problem. People using security controls will be more willing to use it if they are convinced that a problem exists.
2. Likelihood of use. No control is effective unless it is used. However, no control should seriously affect the task being protected.
3. Overlapping controls. Several different controls may apply to address a single vulnerability.
4. Periodic review. Few controls are permanently effective, and judging their effectiveness is therefore an ongoing process.

Questions

- Any Questions?

ITRI 615

Pfleeger Chapter 2 & 12

Whitman & Mattord Chapter 8

Cryptography terminology

- Encryption is the process of encoding a message so that its meaning is not obvious. Also referred to as encode or encipher.
- Decryption is the reverse process. Also referred to as decode or decipher.
- Cryptosystem is a system for encryption and decryption.
- The original form of a message is called plaintext, and the encrypted form is called cyphertext.
- Symmetric encryption uses the same key for encryption and decryption, whereas asymmetric encryption uses a different key for encryption and decryption.

Cryptanalysis (breaking the code)

- A cryptanalyst can attempt to do any or all of six different things:

1. Break a single message.
2. Recognize patterns in encrypted messages, to be able to break subsequent ones.
3. Infer some meaning without even breaking the encryption.
4. Deduce the key, to break subsequent messages easily.
5. Find weaknesses in the implementation or environment of use of encryption.
6. Find general weaknesses in an encryption algorithm.

An algorithm might be theoretically breakable, but impractical to break.

Substitution ciphers

- Caesar cipher is the most well known, as it was first used by Julius Caesar himself.
- A scheme whereby each letter is translated to the letter a fixed number of places after it in the alphabet.
- For example, a Caesar cipher, with a shift of 4 to the right:

  Plaintext: H E L L O
  Ciphertext: L I P P S

- Advantageous that it is quite simple, disadvantage is its obvious pattern.

A secure encryption algorithm

- In 1949 Claude Shannon proposed several characteristics that identify a good cipher.

1. The amount of secrecy needed should determine the amount of labor that's appropriate.
2. The set of keys and algorithm should be free from complexity.
3. The implementation of the process should be as simple as possible.
4. Errors in ciphering should not propagate and cause corruption of further information.
5. The size of the enciphered text should be no larger than that of the original text.

DES (Data Encryption Standard)

- Based on substitution (confusion) and transposition (diffusion).
- Plaintext is encrypted in blocks of 64 bits, and the key can be any 56 bit number, as the extra 8 bits are check digits and do not affect the encryption.
- The user can change the key at any time.
- The plaintext is affected by a series of cycles of a substitution then a permutation.
- Although complex, it is repetitive, thereby making it suitable for implementation on a single-purpose chip (see p. 97 table 2.11).

AES (Advanced Encryption Standard)

- Encryption takes place in blocks of 128 bits.
- Like DES, AES uses repeat cycles.
- 10, 12 or 14 cycles for keys of 128, 192 and 256 bits respectively.
- Each cycle consists of 4 steps (see p. 74 figure 2.9):

1. Byte substitution, according to a substitution table.
2. Shift row, where rows and bytes are shifted depending on key size.
3. Mix column.
4. Add subkey, where a portion of the key is included with the cycle result.

Being new is its major strength.

DES vs AES

- Both DES and AES are examples of symmetrical (secret) key encryption. With secret key the encrypter and the decrypter must have access to the same key to enable communication.

- See table 2.12 on page 100 for a direct comparison between DES and AES.

Public key encryption

- With public key encryption each user has a key that does not have to be kept secret.
- The public nature of the key does not compromise the secrecy of the system.
- The key is therefore divulged, but the trick is to keep the decryption technique secret.
- This goal is accomplished by using two keys, one for encryption and another for decryption.
- See page 104 table 2.13 for a comparison between secret key and public key encryption.

Merkle-Hellman Knapsacks

- The knapsack problem presents a set of positive integers and a target sum, with the goal of finding a subset of the integers that sum to the target.
- Idea is to encode a binary message as a solution to a knapsack problem, reducing the ciphertext to the target sum obtained by adding terms corresponding to 1's in the plaintext.
- The algorithm begins with a knapsack filled with pre-determined values. The plaintext is mapped to these values, where order is extremely important. The values aligned to 1's in the plaintext, are added together to form the target sum.
- The target sum is needed for the decryption, as each set of values in the knapsack sum to a unique target sum.

RSA (Rivest-Shamir-Adelman)

- RSA relies on number theory, where properties of numbers such as their prime factors are studied. Similar to Merkle-Hellman in finding terms that add to a particular sum or multiply to a particular product.
- The algorithm combines results from number theory with the degree of difficulty in determining the prime factors of a given number.
- Based on the underlying problem of factoring large numbers.
- Plaintext block P is encrypted by P^e mod n, with e being the encryption algorithm, and n being a factor chosen as part of the key.
- P is decrypted by $(C^e)^d$ mod n, with C being the ciphertext, and d being the decryption algorithm.

Key exchange

- Used when protected data needs to be sent to someone you don't know and who also doesn't know you. For example e-filing, you don't know who will receive it at SARS and they also don't know you.

- The solution is for the sender to send the receiver:

$$E(k_{PUB\text{-}R}, E(k_{PRIV\text{-}S}, K))$$

where k is the key, both receiver public and private sender, K is the symmetric key, and E is the encryption algorithm (see fig 2.11 p.81).

- The protocol therefore adds two layers of protection, the first is unwrapped with the senders public key, and the second with receivers private key.

Digital signatures

- A protocol that produces the same effect as a real signature.
- A mark that only the sender can make, but other can easily recognize as belonging to the sender.
- A need was realized when money started to be transacted electronically.
- Properties include:

1. It must be unforgeable.
2. It must be authentic.
3. It is not alterable.
4. It is not reusable.

Hash functions

- Mathematical algorithms that generate a message summary or digest to confirm the identity of a specific message and to confirm that there haven't been any changes to the content.
- Created by converting variable-length messages into a single fixed-length value.
- Hash values are computed at the sending and receiving side. These values are compared, and if equal, message transmission has been successful without alteration.
- A hash function will always provide the same hash value for the same message, and the hash value cannot be used to determine message contents.

Questions

- Any questions?

ITRI 615

Pfleeger Chapter 3

Whitman & Mattord Chapter 2

Nonmalicious program errors, buffer overflows

- Programmers often make mistakes, most of which are unintentional and nonmalicious, even though they can still cause serious harm.
- The first type of error is buffer overflows.
- Occurs when the size of the memory allocated to store data is insufficient.
- The attacker can abuse overflows to replace code in the system space. If the attacker can masquerade as the operating system, they can execute many commands in a very powerful role.
- By causing an overflow in the CPU stack register, the attacker can change the stack pointer to point to a block of code the attacker wants.

Nonmalicious program errors Incomplete mediation

- Also referred to as unchecked data values. Typically occurs when either an invalid data type is entered into a variable, for example minute is entered into month, or when a value out of range is entered, for example 1800 as the year.
- Typically handled through exception handling, or by testing values as they are entered.
- Through the unnecessary passing of parameters back to the vendor, the users/attackers have the option to alter the total price, which can work if the required security checks are not performed.
- The amount of damage done depends on the amount of time it goes undetected.

Nonmalicious program errors
Time-of-check to time-of-use errors

- Refers to the errors that occur due to the time delay between verifying access rights and giving access.

- For example, a user requests access to a file, a file which access is allowed for. While the access control mediator checks for access clearance, the user could change the file name descriptor to a file for which they do not have access. Thereby exploiting the time delay.

- The main security implication is ineffective access control, thereby seriously comprising security.

- Prevention includes letting the access checking software own the data until the verification has occurred, or to allow no interruption during the validation.

Malicious programs

- Programs only become security threats when data and state changes trigger it.
- Malicious programs can cause anything from mere irritation to serious harm.
- Documented proof of malicious programs can be found as far back as 1972.
- Malicious code is the general name for unanticipated or undesired effects in programs or program parts.

Kinds of malicious code (Table 3.2 p. 170)

- Virus – a program that can replicate itself and pass on malicious code to other nonmalicious programs by modifying them. Can be transient, where its life depends on its host, or resident, where it locates itself in memory where it can remain active or be activated as a stand alone program.

- Trojan horse – Contains unexpected, additional functionality.
- Logic bomb – Triggers action when condition occurs.
- Time bomb – Triggers action when specified time occurs.
- Trapdoor – Allows unauthorized access to functionality.
- Worm – Propagates copies of itself through a network.
- Rabbit – Replicates itself without limit to exhaust resources.

Virus effects and causes (Table 3.4 p. 193)

Virus effect	How it is caused
Attach to executable program	Modify file directory Write to executable program file
Attach to data or control file	Modify directory, rewrite data Append to data, append data to self
Remain in memory	Intercept interrupt Load self in nontransient memory area
Infect disks	Intercept interrupt, intercept operating system call, modify system file or executable program
Conceal self	Intercept system calls Classify self as hidden file
Spread infection	Infect boot sector, systems program, or ordinary program and its data
Prevent deactivation	Activate before deactivating program Store copy to reinfect

Prevention of virus infection

- Several techniques exist for building a reasonably safe community for electronic contact. They include:

1. Use only commercial software acquired from reliable, well established vendors.
2. Test all new software on an isolated computer.
3. Open attachments only when you know them to be safe.
4. Make a recoverable system image and store it safely.
5. Make and retain backup copies of executable system files.
6. Use virus detectors regularly and update them daily.

Virus truths and misconceptions

- Viruses can infect only Microsoft Windows systems. False.
- Viruses can modify hidden or read only files. True.
- Viruses can appear only in data files, Word documents or programs. False.
- Viruses spread only on disks or through e-mail. False.
- Viruses cannot remain in RAM memory after a power off. True.
- Viruses cannot infect hardware. True.
- Viruses can be malevolent, benign or benevolent. True.

Targeted malicious code

- Trapdoors – an undocumented entry point to a module.
- Salami attack – merges bits of seemingly inconsequential data to yield powerful results.
- Rootkit – a piece of malicious code that goes to great lengths not to be discovered or, if discovered and removed, to reestablish itself whenever possible.
- Privilege escalation – a means for malicious code to be launched by a user with lower privileges but run with higher privileges.
- Interface illusions – a spoofing attack in which all or part of a web page is false.

Targeted malicious code *cont.*

- Keystroke logging – retains a surreptitious copy of all keys pressed.
- Man-in-the-middle attacks – a malicious program that interjects itself between two other programs.
- Timing attacks – using processing time as an indication of when the decryption key value is being approached while guessing.

Developmental controls against program threats

- Software development requires people who can

1. Specify the system.
2. Design the system.
3. Implement the system.
4. Test the system.
5. Review the system.
6. Document the system.
7. Manage the system.
8. Maintain the system.

We must design systems that are both secure and usable. You can't retrofit usable security.
Tools aren't solutions. Mind the upper layers. Keep the customers satisfied. Think and act locally.

Modularity, encapsulation and information hiding

- Three key principles of software engineering are:

- Modularity – creating a design or code in small self-contained units, called components or modules.

- Encapsulation – when components are isolated from the effects of other components, fault tracing is simplified and it is also easier to maintain the system.

- Information hiding – by hiding information, each component hides its precise implementation or some other design decision from the others. This simplifies maintenance and aids with system stability.

Other controls against program threats

- Mutual suspicion – where two programs operate as if other routines in the system were malicious or incorrect.
- Confinement – where a program is strictly limited in what system resources it can access.
- Genetic diversity – some people believe that it is risky having many components of a system come from one source.
- Peer reviews – peer reviews in software engineering can be extraordinarily effective in fault identification and correction.
- Hazard analysis – a set of systematic techniques intended to expose potentially hazardous system states.

Other controls against program threats *cont.*

- Testing – a process activity that hones in on product quality, making the product failure free or failure tolerant.
- Good design – use a philosophy of fault tolerance, have a policy for handling failures, capture the design rationale and history, use design patterns.
- Prediction – predict the risks involved in building and using the system, and mitigate these risks.
- Static analysis – locate and repair security flaws in code before a system is up and running.
- Configuration management – know who is making which changes to what and when.

Other controls against program threats *cont.*

- Learn from mistakes – one of the easiest things that can be done to enhance security.

- Proofs of program correctness – involves making initial assertions about the inputs and then checking to see if the desired output is generated.

Important to note that none of the mentioned controls can guarantee the security or quality of a system.

Questions

- Any questions?

ITRI 615

Pfleeger Chapter 5 Part 1

Whitman & Mattord Chapter 7

Protection in operating systems

Separation is the basis of protection in operating systems. Keeping one user's objects separate from other users.

- Physical separation. Where different physical objects are used, for example printers.
- Temporal separation. Processes with different security requirements are executed at different times.
- Logical separation. Where users operate under the illusion that no other processes exist.
- Cryptographic separation. Where processes conceal their data and computations from outside processes.

Protection in operating systems *cont.*

- An operating system can support separation and sharing in several ways, offering protection at any of several levels.

1. Do not protect.
2. Isolate.
3. Share all or share nothing.
4. Share via access limitation.
5. Share by capabilities.
6. Limit use of an object.

Memory and address protection

- The most obvious problem of multiprogramming is preventing one program from affecting the data and programs in the memory space of other users. Fortunately many protection mechanisms exist which include:

1. Fences. A method to confine users to one side of a boundary (see figure 5.6, p. 298).

2. Relocation. Taking a program written as if it began at address 0 and changing all addresses to reflect the actual address at which the program is located in memory.

Memory and address protection *cont.*

3. Base/bounds registers. Base (or fence) registers provide a starting address where programs will be loaded for execution, but no upper one. Bounds registers are upper address limits.

4. Tagged architecture. Every word of machine memory has one or more extra bits to identify the access rights to that word. These access bits can be set only by privileged operating system instructions.

5. Segmentation. Involves the simple notion of dividing a program into separate pieces.

6. Paging. Program is divided into equal-sized pieces called pages, and memory is divided into equal-sized units called page frames. Paging and segmentation can also be combined.

Control of access to general objects

- Protecting objects is a more general problem than protecting memory.
- There are several complementary goals in protecting objects, namely:

1. Check every access. Every access by a user to an object should be checked.
2. Enforce least privilege. A subject should have access to the smallest number of objects necessary to perform some task.
3. Verify acceptable usage. After access has been granted, the intended action to be performed should also be verified.

Control of access to general objects *cont.*

- Several object access control mechanisms exist, including:

1. Directories. A file directory that has a unique owner who is able to grant and revoke access rights. This directory lists all the files to which that user has access.

2. Access control list. One list for each object, indicating who has access to it and what access they have.

3. Access control matrix. A table in which each row represents a subject, each column represents an object, and each entry is the set of access rights for that subject to that object.

4. Capability. A capability is a ticket giving permission to a subject to have a certain type of access to an object.

Control of access to general objects *cont.*

5. Kerberos. Implements both authentication and access authorization by means of capabilities, or tickets, secured with symmetric encryption. Requires two systems, the authentication server and the ticket-granting server.

6. Procedure-oriented access control. A procedure that controls access to objects, in addition to the protection provided by the operating system.

7. Role-based access control. Lets us associate privileges with groups, i.e. administrators vs. standard users.

File protection mechanisms

As the number of users increase, so too does the complexity of file protection schemes.

1.All-none protection. In early operating systems all files were public. Users could perform any action on anybody's file. The only "protection" involved trust along with ignorance. It was felt that you would treat the files of other users as you would like yours to be treated. Only option was passwords, if needed.

It was unacceptable for several reasons.

- Lack of trust.
- Too coarse.
- Rise of sharing.
- Complexity.
- File listings.

File protection mechanisms *cont.*

2. Group protection. Focused on identifying groups of users who had some common relationship. Group members then had access to the files of only the other members of the group. It therefore overcame some of the shortcomings of all-none protection, but caused some new difficulties.

- Group affiliation.
- Multiple personalities.
- All groups.
- Limited sharing.

File protection mechanisms *cont.*

Individual permissions was the next attempt to improve file protection.

3. Persistent permission. A mechanism which uses access lists and tokens or passwords to control access. Revocation of access is relatively difficult.

4. Temporary acquired permission. Based on a three-level user-group-world hierarchy. Utilizes a permission called **set userid** (suid). If this protection is set for a file to be executed, the protection level is that of the file's owner, not the executor. For example, suppose Tom owns a file and allows Ann to execute it with suid. When Ann executes the file, she has the protection rights of Tom, not of herself.

User authentication

- An operating system bases much of its protection on knowing who a user of the system is.

- Authentication mechanisms use any of three qualities to confirm a user's identity. These qualities are:

1. Something the user knows.

2. Something the user has.

3. Something the user is.

User authentication *cont.*

- Passwords are mutually agreed-upon code words, assumed to be known only to the user and the system. They are the most common authentication mechanism for user to operating system.

- They do however have some difficulties.

1. Loss.
2. Use.
3. Disclosure.
4. Revocation.

- Additional authentication information can also be used, for example office hours, IP addresses, etc. This is referred to as multifactor authentication.

User authentication *cont.*

- Attacks on passwords can take many forms. Listed below, in decreasing order of difficulty, are ways which might be tried to obtain a user's password.

1. Try all possible passwords.
2. Try frequently used passwords.
3. Try passwords likely for the user.
4. Search for the system list of passwords.
5. Ask the user.

- Loose-lipped systems. More often than not password authentication systems are their own enemies. For example, entering an invalid user name usually leads to an "invalid user name" response, similarly an incorrect password.

User authentication *cont.*

- Exhaustive (brute force) attack. The attacker tries all possible passwords, usually in some automated fashion. The number of possibilities depends on the particular computing system.

- Probable passwords. Think of a word. Is it long, uncommon, or hard to pronounce or spell? Penetrators looking for passwords exploit these human traits.

- Passwords likely for a user. We tend to choose our passwords according to something that is meaningful to us.

User authentication *cont.*

- Plaintext system password list. To validate passwords, the system must compare them with actual passwords. On some systems the password list is a file, containing two columns with user ID's and corresponding passwords. Merely protecting this table is often insufficient.

- Encrypted password file. Similar to the plaintext list, with the only difference being the fact that the ID's, Passwords, or both are encrypted. When a password is entered, the comparative data is decrypted, and then compared to check for validity.

- Indiscreet users. As users tend to forget passwords or might be just plain lazy, they tend to keep record of their passwords on or around their computers. Research has shown that users sometimes even tend to simply divulge their passwords, for no obvious reason.

User authentication *cont.*

- Several guidelines have been developed for password selection.

1. Use characters other than just A – Z.
2. Choose long passwords.
3. Avoid actual names or words.
4. Choose an unlikely password.
5. Change the password regularly.
6. Don't write it down.
7. Don't tell anyone else.

User authentication *cont.*

- The current trend in user authentication is moving towards biometrics. Biometrics are biological authenticators based on some physical characteristic of the human body. Current technologies include fingerprints, hand geometry, retina and iris, voice, handwriting, blood vessels in the finger and face.

- There are however several problems with biometrics.

1. Relatively new, and considered by some to be intrusive.
2. Recognition devices are costly.
3. Readers use sampling to decide on being close enough.
4. Can become a single point of failure.
5. There are still false readings.
6. The speed required limits accuracy.
7. Although we think biometrics are unique, forgeries are possible.

Questions

- Any questions?

ITRI 615

Pfleeger Chapter 5 Part 2

A trusted operating system

We say that an operating system is trusted if we have confidence that it provides

1.Memory protection,

2.File protection,

3.General object access control, and

4.User authentication

Consistently and effectively.

A trusted operating system *cont.*

The 4 major underpinnings of a trusted operating system are:

1.Policy. Statements of what the system should do and how it should do it.

2.Model. A representation of the policy the operating system will enforce. I.e. the proposed system has to meet its requirements while protecting appropriate objects and relationships.

3.Design. Involves both what the trusted operating system is and how it is to be constructed.

4.Trust. Rooted in two aspects, features (has all the necessary functionality) and assurance (confident that it will enforce the security policy correctly).

Trusted software

- We say that software is trusted software if we know that the code has been rigorously developed and analyzed, giving us reason to trust that the code does what it is expected to do and nothing more.

- Trust is based on looking for certain key characteristics:

1. Functional correctness. Program does what it is supposed to do.
2. Enforcement of integrity. Maintains the correctness of the data even with invalid commands.
3. Limited privilege. Access to secure data is minimized and not passed on.
4. Appropriate confidence level. Rated at a degree of trust appropriate for the kind of data and environment in which it will be used.

Trusted operating system design

- Good design principles are always good for security.

- Several important design principles that are quite particular to security and essential for building a solid, trusted operating system include:

1. Least privilege. Each entity should operate using the fewest privileges possible.

2. Economy of mechanism. The design of a protection system should be small, simple and straightforward.

3. Open design. The mechanism should be public, depending on the secrecy of relatively few key items.

Trusted operating system design *cont.*

4. Complete mediation. Every access attempt must be checked.

5. Permission based. The default should be denial of access, rather than the other way round.

6. Separation of privilege. Access to objects should depend on more than one condition.

7. Least common mechanism. Shared objects provide potential channels for information flow. Physical or logical separation reduce the risk from sharing.

8. Ease of use. If a mechanism is easy to use, it is unlikely to be avoided.

Security features of trusted operating systems

- The key security features of a trusted operating system include:

1. User identification and authentication.
2. Mandatory access control.
3. Discretionary access control.
4. Object reuse protection.
5. Complete mediation.
6. Trusted path.
7. Audit.
8. Audit log reduction.
9. Intrusion detection.

Assurance in trusted operating systems

- Typical operating system flaws can be attributed to four main known vulnerabilities:

1. User interaction. The largest single source.

2. Ambiguity in access policy. We want to separate users and their resources, but more often than not these users depend on shared resources.

3. Incomplete mediation. When a requested access is only authorized once per user interface operation, process execution or machine interval.

4. Generality. Operating systems tend to be too general to allow interoperability with software from other vendors.

Assurance in trusted operating systems *cont.*

- Once we understand the potential vulnerabilities in a system, we can apply assurance techniques to seek out the vulnerabilities and mitigate or eliminate their effects.

- The three available techniques are testing, verification and validation.

1. Testing. The most widely accepted assurance technique. Conclusions from testing are based on the actual product being evaluated, but unfortunately testing is almost always constrained by a project's budget and schedule. A testing strategy often used in computer security is called penetration testing, tiger team analysis, or ethical hacking.

Assurance in trusted operating systems *cont.*

2. Verification. The most rigorous method of analyzing security. Uses rules of mathematical logic to demonstrate that a system has certain security properties. Verification confirms that the operating system provides the security features it should and nothing else. The two principal difficulties of verification methods are time and complexity.

3. Validation. A more general approach to ensuring correctness. The counterpart of verification which assures that the system developers have implemented all requirements. Validation of an operating system can take place through either:

- Requirements checking.
- Design and code reviews.
- System testing.

Open source software

- What are the advantages of open source for computer security?

- What are the disadvantages of open source for computer security?

- Additional benefits of open source:

1. Cost. As the source code is available to the public, high fees will simply lead to unofficial trading.
2. Quality. The code can be analyzed by many independent reviewers.
3. Support. As the public finds flaws, they are also in the best position to suggest fixes for the flaws.
4. Extensibility. Code can easily be extended to meet new needs, and the new code can also be easily distributed.

Questions

- Any questions?

ITRI 615

Whitman & Mattord Chapter 9

Physical Access Controls

- In facilities management, a secure facility is a physical location that has in place controls to minimize the risk of attacks from physical threats (see Amy Windahl was back early from lunch p. 467 and offline guard duty, p. 472).

- Some physical access controls that can aid with securing a facility include:

1. Walls, fencing and gates.

2. Guards.

3. Dogs.

4. ID cards and badges.

Physical Access Controls *cont.*

5. Locks and keys.
6. Mantraps. See figure 9.3, p. 476.
7. Electronic monitoring.
8. Alarms and alarm systems.
9. Computer rooms and wiring closets.
10. Interior walls and doors.

Fire security and safety

- The most serious threat to people in an organization is fire.
- It accounts for more property damage, personal injury, and death than any other threat to physical security.
- Fire suppression systems are devices that are installed and maintained to detect and respond to a fire, potential fire, or combustion danger situation.
- The flame point, or temperature of ignition, depends on the material and can be as low as a few hundred degrees.

Fire security and safety *cont.*

- Fire detection can be either manual or automatic. Manual typically through human responses, and automatic through devices.

- There are three types of automatic fire detection systems.

1. Thermal detection systems. Contains a heat sensor that reacts either to a high temperature or to a rapid rise in temperature.

2. Smoke detection. Based on photoelectric sensors, ionization sensors, or air-aspirating detectors.

3. Flame detectors. Detects the infrared or ultraviolet light produced by an open flame.

Fire suppression

- Can consist of portable, manual or automatic apparatus.
- Portable extinguishers are used for smaller fires and where fixed apparatus is impractical.
- Portable extinguishers are rated by the type of fire they can combat, as follows:

1. Class A. Fires that involve ordinary combustible fuels.
2. Class B. Fires fueled by combustible liquids or gases.
3. Class C. Fires with energized electrical appliances or equipment.
4. Class D. Fires fueled by combustible metals.

Fire suppression *cont.*

- Manual and automatic systems are those designed to apply suppressive agents. They are usually either sprinkler or gaseous systems.

- Sprinkler are divided into three implementations, namely:

1. Wet-pipe. Contains pressurized water in all pipes.

2. Dry-pipe. Contains pressurized air, which is released before the water. Reduces the risk of accidental leakage.

3. Pre-action. Two phase process whereby stage one involves filling the pipes with water, and stage two then involves manual activation of individual sprinklers.

Fire suppression *cont.*

- Gaseous emission systems are often used to protect chemical and electrical processing areas, as well as facilities that house computing systems (see figure 9.5 p. 519).
- Can be either self-pressurizing or pressurized with an additional agent.
- Only two major types, using either carbon dioxide or Halon.
- Unlike carbon dioxide, Halon does not extinguish life forms that are dependent on oxygen as well. It is however an ozone-depleting substance, and is no longer used in suppression systems. Many alternatives are available.

Failure of supporting utilities

- Failure of supporting utilities can have a significant impact on the safe operation of a facility. Each of these utilities must be properly managed in order to prevent damage to information and information systems. These utilities include:

1.Heating, ventilation, and air conditioning.

2.Temperature and filtration.

3.Humidity and static electricity.

4.Ventilation shafts.

Failure of supporting utilities *cont.*

5. Power management and conditioning.
6. Grounding and amperage.
7. Uninterruptible power supply.
8. Emergency shutoff.
9. Water problems.
10. Structural collapse.
11. Maintenance of facility systems.

Interception of data

- Direct observation. Requires that an individual be close enough to the information to breach confidentiality. Major risk is when information is removed from a secure facility.

- Interception of data transmissions. When attackers can access the media transmitting the data, they don't have to be close to the source. Includes tapping into a LAN, eavesdropping over a wireless LAN, etc.

- Electromagnetic interception. Eavesdropping on the electromagnetic signals emitted when electricity moves through cables. Has been hotly debated whether this is actually possible, despite the fact that the US government and military are investing a good deal of money in securing their systems from EM interception.

Questions

- Any questions?